INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Exploration Rapide et Exhaustive des Banques d'ADN

Pascale Guerdoux-Jamet, Dominique Lavenier

N° 2580

Juin 1995

PROGRAMME 1

ISSN 0249-6399

INRIA
RENNES

Exploration Rapide et Exhaustive des Banques d'ADN

Pascale Guerdoux-Jamet, Dominique Lavenier

Programme 1 — Architectures parallèles, bases de données, réseaux et systèmes distribués
Projet API

Rapport de recherche n° 2580 — Juin 1995 — 36 pages

Résumé : Ce rapport présente un filtre matériel qui, intégré dans des logiciels couramment utilisés pour l'exploration des banques de séquences d'acides nucléiques, minimise deux inconvénient majeurs : la non exhaustivité des solutions et la lenteur des calculs. La substitution des parties logiciels critiques par ce dispositif matériel réduit les temps de calculs dans de très fortes proportions (de quelques heures à quelques minutes) tout en garantissant des résultats rigoureusement exactes. La mise en oeuvre étudiée (et testée sur des composants reprogrammables électriquement) propose, dans un faible volume, un accélérateur matériel qui peut aisément s'insérer dans un slot d'extension d'un PC ou d'une station de travail.

Mots-clé : comparaison de séquences d'ADN, filtre systolique

(Abstract: pto)

Ce travail est supporté par le GREG, Groupement de Recherches et d'Etudes sur les Génomes

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : (33) 99 84 71 00 – Télécopie : (33) 99 84 71 71

Fast and Exhaustive Scan of DNA Banks

Abstract: This report presents a systolic filter for speeding up the scan of DNA databases. The filter acts as a co-processor which performs the more intensive computations occurring during the scan. Validation has been done on a based FPGA prototype board tightly connected to a workstation and has shown that the filter may boost considerably the performances of current workstation.

Key-words: DNA comparison, systolic filter

1 Introduction

L'exploration[1] des banques de séquences d'ADN est un des traitements de base de la biologie moléculaire ; cette opération consiste, entre autre, à confronter une séquence particulière à une ou plusieurs banques connues : le but est de rechercher des similitudes avec des séquences déjà répertoriées.

Le temps de calcul de ces traitements dépend à la fois de la taille des banques, des séquences à traiter, de la puissance des machines informatiques et de la sensibilité demandée. Sur une station de travail (une SPARC-2, par exemple), ce temps de calcul peut varier de quelques minutes à plusieurs heures.

Dans les années futures, le temps de calcul de l'exploration des banques d'ADN devrait rester plus ou moins constant. En effet, la puissance des ordinateurs augmente pratiquement au même rythme que la taille des banques d'ADN. La courbe (trait plein) de la figure 1 montre la progression en taille, depuis 1986, d'une banque très utilisée, la banque GenBank [4]. L'évolution de cette banque est représentative de l'évolution de l'ensemble des banques de séquences biologiques. On estime qu'elles s'accroissent annuellement d'un facteur compris entre 1,4 et 1,5. Et aujourd'hui, rien ne laisse présager une rupture dans cette progression.

Parallèlement, ces dernières années ont vu croitre, de manière constante, la puissance des ordinateurs. Cette augmentation résulte principalement des progrès réalisés dans le domaine des microprocesseurs : forte densité d'intégration, fréquence d'horloge élevée, nouvelles architectures (RISC, superscalaire et/ou superpipeline), etc. A titre d'exemple, la courbe (pointillé) de la figure 1 montre la puissance, exprimée en MIPS (Million d'Instructions Par Seconde), des processeurs de la famille 80x86 lorsqu'ils ont été introduits sur le marché [11]. Il s'agit de la puissance initiale, puissance qui, par la suite et pour un processeur donné, s'est accrue en fonction des progrès technologiques.

Les deux courbes suivent quasiment la même progression. Ainsi, comme la quantité de calculs requise pour explorer les banques est directement proportionnelle à leur taille, on peut estimer que les performances des ordinateurs futurs compenseront – plus ou moins – le volume de calculs requis pour explorer les banques de demain.

1. traduction du terme anglosaxon *scan*

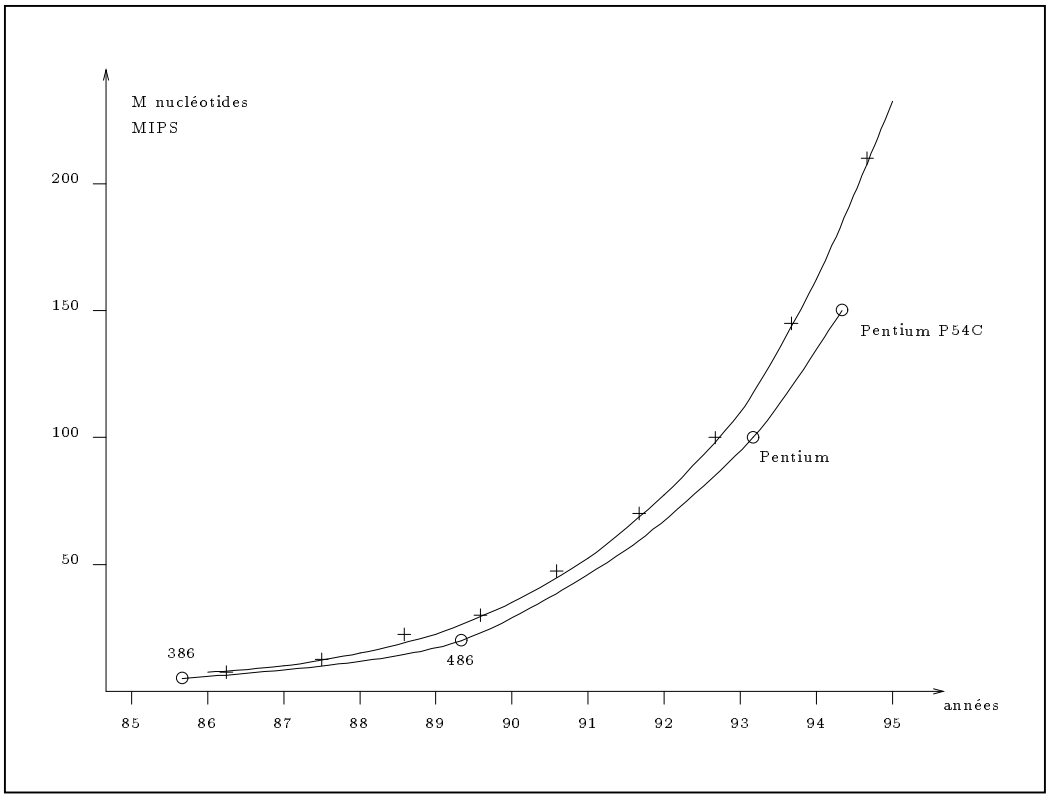

FIG. 1 - *évolution de la banque GenBank en million de nucléotides (trait plein) et de la puissance des processeurs de la famille 80x86 exprimée en* MIPS *(pointillé)*

Les logiciels couramment utilisés, tels que BLASTN [1] ou FASTA [14] – deux logiciels de référence –, pour rechercher des similitudes entre une séquence particulière (la séquence test) et une banque ont été concus, en grande partie, pour limiter au maximum le temps d'exploration. Le principe est principalement basé sur des heuristiques algorithmiques axées sur la compression des séquences.

Ces heuristiques sont controlées par des paramètres qui permettent soit un temps de calcul court, mais avec des résultats approchés, soit des résultats exacts, mais avec un temps de calcul très long. C'est à l'utilisateur – lorsqu'il le peut – de trouver le meilleur compromis vitesse d'exécution/qualité des résultats. La réduction du temps de calcul se fait généralement au détriment de la qualité des résultats.

En général, les logiciels fonctionnent en deux temps : une première étape est consacrée à la détection des régions potentiellement intéressantes (d'un point de vue similarité). Dans un second temps, ces régions sont réévaluées précisément pour une localisation précise et un affinement de la mesure de similarité.

Dans tous les cas, la phase critique est la détection rapide de régions qui présentent des similitudes potentielles. C'est donc cette phase qui inclut les heuristiques permettant de restreindre l'exploration. Par conséquent, elle conditionne à la fois la sensibilité de la détection et le temps d'exécution. Nous proposons de remplacer cette étape par un accélérateur matériel qui s'appuie sur un algorithme de détection exhaustif : toutes les solutions sont systématiquement considérées. Les résultats produits sont donc optimaux.

D'un point de vue implémentation matérielle, cet algorithme possède un atout important : sa *régularité* ; cette caractéristique permet une parallélisation efficace du calcul, et par conséquent des temps d'exécution courts. L'architecture proposée est un réseau linéaire composé de processeurs cablés (non programmables) réalisant plusieurs opérations simultanément. Le parallélisme se situe donc à la fois en l'exécution d'une succession d'opérations en une seule étape – au sein d'un processeur – et en l'exécution simultanée du calcul sur l'ensemble des processeurs.

Notre approche a été validée par une réalisation matérielle. L'implémentation a été effectuée sur un support à base de logique reconfigurable : la carte Perle-1 développée par le laboratoire Parisien de DEC [5] [6]. Il s'agit d'une

carte expérimentale regroupant une matrice de composants Xilinx et connectée à une station de travail via une interface rapide (100 Moctets/sec).

Les résultats obtenus montrent que l'usage de cet accélérateur matériel permet d'obtenir en des temps très raisonables (quelques dizaines de secondes) des résultats complets, comparables à ceux de FASTA ou BLASTN (version 1.4) utilisés avec une sensibilité maximale.

La suite de ce document est structurée de la manière suivante : le prochain paragraphe décrit l'algorithme de détection de segments similaires. Le principe est celui utilisé dans BLASTN et FASTA, deux logiciels que nous analysons ensuite (paragraphe 3) sous l'angle sensibilité/temps d'exécution. Les deux paragraphes suivants montrent l'architecture de l'accélérateur matériel et la mise en oeuvre qui en a été faite. Des résultats de sensibilité et d'accéleration sont ensuite présentés avant de conclure sur une comparaison par rapport à d'autres systèmes existants et d'énoncer quelques perspectives de ces travaux.

2 Détection des segments similaires

Dans cette partie, nous présentons l'algorithme permettant de localiser les portions similaires entre deux séquences. Cet algorithme est le pillier commun aux logiciels d'exploration de banques de séquences d'ADN lorsqu'il s'agit de répérer des alignments sans insertions ni omissions, même s'il est souvent adapté à des heuristiques réduisant le temps de calcul.

Déterminer une zone de similitude entre deux séquences consiste à mettre en correspondance tous les caractères d'une séquence avec l'autre, puis à calculer un indice de ressemblance. Considérons, par exemple, la séquence $S0 =$ `CACGT` et la séquence $S1 =$ `TTACGCT` ; les dix possibilités de mise en correspondance sont les suivantes :

```
1 -        C ACGT          6 -    CACGT
      TTACG C                       TTACG C

2 -        CA CGT          7 -    CACGT
       TTAC GC                       TTAC GC

3 -        CAC GT          8 -    CACGT
        TTA CGC                       TTA CGC
```

```
4 -          CACG T          9 -    CACGT
           TT ACGC                     TT ACGC

5 -          CACGT          10 -    CACGT
            T TACGC                     T TACGC
```

Une mise sous forme matricielle donne une représentation plus concise; chaque diagonale de la matrice représente la mise en correspondance de deux sous-séquences (alignement):

```
     C  A  C  G  T

T    6  7  8  9 10
T    5  6  7  8  9
A    4  5  6  7  8
C    3  4  5  6  7
G    2  3  4  5  6
C    1  2  3  4  5
```

Les valeurs de la matrice représentent le numéro de la diagonale. Par exemple, la diagonale 7 représente la mise en correspondance de la sous-séquence `ACGT` de `S0` et de la sous-séquence `TTAC` de `S1`.

Il s'agit maintenant, pour chaque diagonale, de déterminer des zones similaires. Le principe est de calculer un score en fonction du nombre de paires de caractères consécutifs identiques. L'algorithme consiste à incrémenter le score si les caractères sont égaux ou à le décrémenter lorsqu'ils sont différents, sauf si le score devient négatif ou nul, de manière à le maintenir supérieur (ou égal) à zéro. Toujours sur le même exemple, à partir d'un score nul pour chaque diagonale et en ajoutant ou retranchant 1, on obtient la matrice suivante:

```
     C  A  C  G  T

T    0  0  0  0  1
T    0  0  0  0  1
A    0  1  0  0  0
C    1  0  2  0  0
G    0  0  0  3  0
C    1  0  1  0  2
```

calculée à l'aide de la double boucle :

```
for (i=1; i<=n0; i++)
  for (j=1; j<=n1; j++)
    {
      M[i][j]=M[i-1][j-1] + ((S0[i]==S1[j]) ? 1 : -1);
      if (M[i][j] < 0) M[i][j] = 0;
    }
```

Les zones similaires sont détectées par analyse des valeurs de la matrice, notamment en répérant les scores importants. Rappelons que pendant la première phase on veut simplement obtenir une information partielle sur les zones de similitude ; elle se résume au score maximum calculé pour chaque diagonale. Ainsi, l'information "la diagonale 5 contient un score maximum de 3" sera simplement délivrée. Elle signifie qu'entre les sous séquences `CACGT` et `TACGC` il existe une ou plusieurs zones présentant, en première approximation, une *certaine* similitude. Cette information, si elle a été retenue, sera traitée par la suite pour recalculer précisément le score et l'emplacement des segments similaires.

Une instruction mémorisant la valeur maximale de chaque diagonale est alors insérée dans la boucle de calcul :

```
for (i=1; i<=n0; i++)
  for (j=1; j<=n1; j++)
    {
      M[i][j]=M[i-1][j-1] + ((S0[i]==S1[j]) ? 1 : -1);
      if (M[i][j] < 0) M[i][j]=0;
      if (MAXDIAG[n0-i+j] < M[i][j]) MAXDIAG[n0-i+j]=M[i][j];
    }
```

Après l'exécution de cette double boucle, le tableau `MAXDIAG` contient les scores maximum des $n0 + n1 - 1$ diagonales. Puisqu'on désire simplement conserver le score maximum de chaque diagonale de la matrice, il n'est plus nécessaire de mémoriser entièrement cette dernière. La double boucle devient :

```
for (i=1; i<=n0; i++)
  for (j=1; j<=n1; j++)
    {
      D[n0-i+j] = D[n0-i+j] + ((S0[i]==S1[j]) ? 1 : -1);
      if (D[n0-i+j] < 0) D[n0-i+j]=0;
      MAXDIAG[n0-i+j] = max(MAXDIAG[n0-i+j],D[n0-i+j]);
    }
```

Plus généralement, le score est calculé à partir de deux valeurs, M et N, qui représentent respectivement les coûts d'une identité (match) et d'une non-identité (mismatch). M correspond à une valeur positive tandis que N correspond à une valeur négative. L'algorithme devient :

```
for (i=1; i<=n0; i++)
  for (j=1; j<=n1; j++)
    {
      D[n0-i+j] = D[n0-i+j] + ((S0[i]==S1[j]) ? M : N);
      if (D[n0-i+j] < 0) D[n0-i+j]=0;
      MAXDIAG[n0-i+j] = max(MAXDIAG[n0-i+j],D[n0-i+j]);
    }
```

La détection des segments similaires est essentiellement basée sur l'exécution de cette double boucle. Si p est le nombre de séquences d'une banque, n_t la longueur de la séquence test et n_b la longueur moyenne des séquences dans la banque, le temps de calcul t_d consacré à la détection est donné par :

$$t_d = p \times n_t \times n_b \times t_b$$

où t_b représente le temps d'exécution du corps de la boucle interne. En fait, le temps consacré à cette phase de détection constitue la quasi totalité du temps d'exécution total, les phases suivantes (mises en formes, localisation exacte des segments sur la diagonale, calcul de probabilités, etc) étant très peu couteuses en calculs. Aussi, les logiciels d'exploration de banques de séquences biologiques ont-ils principalement axés lors efforts sur la diminution du temps passé dans cette phase de détection.

Le paragraphe suivant présente les logiciels BLASTN et FASTA ; tous les deux ont recours à des heuristiques pour limiter le temps passé dans cette étape du calcul.

3 Analyse des logiciels FASTA et BLASTN

3.1 FASTA

FASTA répond à la question : quelles sont les séquences de la banque qui *ressemblent* le plus à une séquence test donnée ? Le résultat de cette interrogation est une suite de séquences classées suivant le degré de similitude. La structure (simplifiée) du logiciel est la suivante :

```
Acquisition de la sequence test;
Pour chaque sequence de la banque
 {
   - Detection approximative des zones similaires
     avec la sequence test;
   - Localisation precise des zones similaires et
     calcul d'un score;
   - Alignement;
 }
Impression des resultats;
```

Le code est articulé autour d'une boucle principale qui parcourt, séquence par séquence, l'ensemble de la banque. Pour chaque paire de séquences (séquence test, séquence banque), on détermine d'abord des zones potentiellement similaires, zones qui sont ensuite précisément réévaluées. A ce stade du calcul on retient les 10 meilleurs zones ; elles servent ensuite à produire un alignement global entre les deux séquences. Celui-ci est construit en connectant les zones détectées par programmation dynamique (l'insertion ou l'omission de caractères est autorisée) ; l'alignement final tente de mettre en correspondance la plus courte des deux séquences (la séquence test ou la séquence de la banque) avec une portion de l'autre. L'impression des résultats consiste en une suite d'alignements de la séquence test avec les N séquences de la banque jugées les plus ressemblantes, N étant un paramètre positionné par l'utilisateur.

FASTA a recours à une heuristique principale pour réduire de manière très importante le volume de calculs (donc le temps de calcul). Il introduit la notion de *ktup* : le terme *ktup* indique un regroupement de plusieurs caractères en un seul. Par exemple, un *ktup* de 4 transpose l'alphabet de départ (4 lettres : A,C,G,T) en un alphabet de 256 lettres (4^4). Le but est de raccourcir les

séquences, en les recodant avec ce nouvel alphabet, de manière à limiter le nombre d'itération de la double boucle de détection. L'accélération produite par cette technique est remarquable. L'inconvénient majeur est que plus le *ktup* est grand, moins la recherche est efficace.

L'analyse du code a montré que la majeure partie du temps de cacul est consacrée à la détection des segments similaires, surtout lorsqu'une sensibilité maximale était demandée. En fait, cette partie fonctionne comme un filtre : elle a pour rôle de ne retenir que les alignements susceptibles de donner de bons résultats. Et pour atteindre ce but, il faut, pour chaque séquence de la banque, essayer de mettre en correspondance toutes les sous-séquences possibles de la séquence test et calculer le degré de similitude correspondant à chaque alignement. Le volume de calcul est considérable.

Les traitements consistant, d'une part, à évaluer plus précisément si ces alignements sont effectivement intéressants et, d'autre part, à combiner les alignements retenus pour produire l'alignement final représentent peu de calculs. En effet, même si ces calculs sont plus complexes, ils portent sur un volume de données restreint comparé au volume de données initial. Par exemple, pour des tailles de séquence de l'ordre de 1000, plus de 98 % du temps total d'exécution du programme est consacré au filtrage. Il est alors clair que l'accélération du code doit porter sur cette partie.

3.2 BLASTN

BLASTN (version 1.3) recherche également le degré de similitude entre les séquences d'une banque et une séquence test. Le résultat est une liste de séquences dont au minimum une région dépasse un seuil de similarité (S) avec la séquence test. On note HSP (*High-scoring Segment Pair*), tout segment entre deux séquences dont le degré de similitude est supérieur ou égal à S.

La structure (simplifiée) du logiciel est la suivante :

```
Acquisition de la sequence test;
Pour chaque sequence de la banque
  {
    - Detection approximative des regions de similarite entre
      chaque paire de sequences;
    - Extension locale de ces regions et calcul d'un
      score de similarite;
    - Selection des sequences de la banque ayant au moins un HSP;
  }
Impression des resultats;
```

La banque est parcourue séquence par séquence en cherchant, pour chaque paire (séquence test, séquence banque), les régions potentiellement intéressantes. Le critère retenu est la présence d'au moins 12 identités de symboles consécutives entre les deux séquences.

Ces régions de 12 caractères identiques sont, par la suite, étendues à droite et à gauche afin de calculer le score final de similitude de chacune d'entre elle. L'impression des résultats correspond à la liste des séquences de la banque qui ont un score qui a dépassé S : c'est la liste des HSP.

BLASTN 1.3 a donc recours à un double filtrage pour le choix des séquences "ressemblantes" : le premier a lieu au moment de la recherche des régions potentiellement intéressantes. Il permet de réduire de manière drastique le nombre de régions à étendre (donc le temps de calcul) ; mais cela se paye en sensibilité ! En effet, on peut très bien avoir une séquence qui présente une ressemblance intéressante avec la séquence test mais sans que l'on ait 12 identités consécutives. Cette séquence est écartée par le premier filtre. D'un point de vue biologique, cela pose le problème de ne récupérer que les séquences qui se ressemblent sur une grande portion plutôt que celles qui sont similaires par plusieurs petits fragments entrecoupés de non-identités. On sait pourtant que les deux types de similarité sont aussi informatifs pour l'analyse d'une séquence.

Le premier filtre de BLASTN 1.3 permet donc d'avoir une grande rapidité dans les calculs mais une non-exhaustivité des résultats. Le second filtre

consiste uniquement à ne conserver que les séquences de la banque qui présentent au moins un HSP (score au dela du seuil S).

Cette heuristique de recherche par mot de 12 symboles fixe le compromis temps de calcul/sensibilité. Seul le second filtre peut être ajusté en fonction de la valeur du seuil S choisie. Le logiciel est rapide, mais les résultats peuvent être incomplets au regard de ceux obtenus avec FASTA utilisé avec une sensibilité maximale ($ktup$=1).

Il est clair que dans le cas de BLASTN 1.3, le biologiste s'intéressera à une plus grande exhaustivité des résultats plutôt qu'à une accélération du temps de calcul pendant la phase de filtrage.

Une version récente de BLASTN, la version 1.4, permet une meilleure sensibilité en autorisant, comme dans FASTA, une recherche caractère par caractère au lieu d'un filtrage initial par bloc de 12 caractères. Les résultats deviennent alors très satisfaisants, mais au détriment du temps de calcul qui devient excessif.

3.3 Conclusion

Compte tenu de l'analyse des deux logiciels, le remplacement de la phase de filtrage par un accélérateur matériel assurant cette fonction permettrait pour :

- FASTA et BLASTN 1.4 : d'amméliorer le temps de calcul, surtout en situation de sensibilité maximale.
- BLASTN 1.3 : d'obtenir des résultats plus complets.

Le paragraphe suivant montre comment réaliser, sur une structure matérielle, la fonction de détection de segments similaires.

4 Filtre systolique

L'analyse des dépendances de la double boucle indique que le calcul de chaque diagonale est indépendant. L'ensemble de ces calculs peut donc, potentiellement, être effectué en parallèle.

La parallélisation consiste à associer un processeur à chaque diagonale. Le traitement d'un processeur est celui décrit dans le corps de la double boucle,

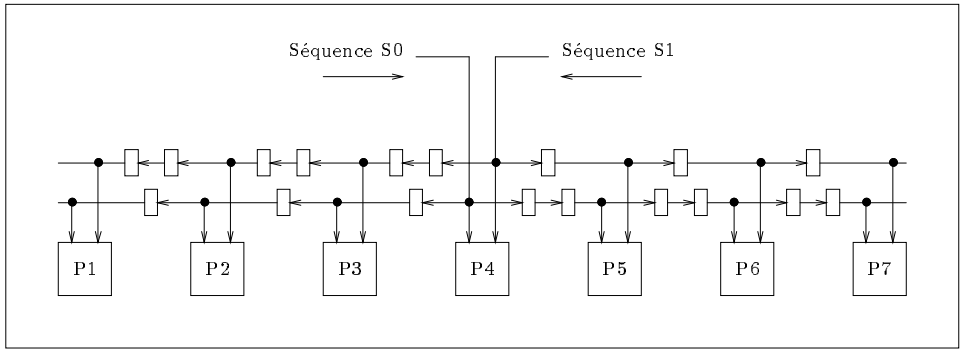

FIG. 2 - *réseau systolique linéaire pour le calcul des scores maximum des diagonales*

c'est à dire un calcul de maximum entre un score courant (`D`) et un maximum précédemment établit (`MAXDIAG`). Le score courant est évalué par rapport aux caractères des séquences : si les caractères sont identiques, le score est incrémenté d'une valeur `M` ; s'ils sont différents le score est diminué d'une valeur `N`, tout en le maintenant positif ou nul.

Les données sur lesquelles un processeur travaille sont fonction du numéro de la diagonale qu'il traite ; en d'autres termes, chaque processeur doit posséder les deux sous-séquences correspondant à la diagonale considérée. En fait, la répartition des données peut être réalisée suivant une approche systolique (cf annexe 1) ; cela évite, pour chaque processeur, de mémoriser en permanence l'ensemble des données qui lui est nécessaire : il reçoit des données de ses voisins, les utilise, puis les propage vers d'autres voisins.

La structure systolique retenue consiste en un réseau linéaire où chaque processeur est connecté suivant le schéma de la figure 2. Deux voies distincts acheminent les caractères des deux séquences vers les processeurs. Les registres insérés sur ces chemins mémorisent temporairement une portion des séquences ; ils permettent une répartition temporelle et spatiale des caractères sur l'ensemble des processeurs.

Considérons, par exemple, les 2 séquences `S=TCGT` et `S=ACGT` et leur émission vers le réseau de la figure 2, à raison d'un caractères par *top* d'horloge. En fonction du temps, chaque processeur verra arriver sur ces entrées les ca-

ractères suivants :

temps	P1	P2	P3	P4	P5	P6	P7
1				T A			
2			T	C C	A		
3		T	C A	G G	T C	A	
4	T	C	G C	T T	C G	C	A
5	C	G A	T G		G T	T G	C
6	G	T C	T		T	C T	G
7	T A	G				G	T T

Les caractères qui arrivent sur le processeur Pi correspondent aux caractères impliqués dans le score de la diagonale Pi.

5 Mise en oeuvre

L'implémentation du filtre systolique a été réalisée sur une carte expérimentale à base de logique reconfigurable. Il s'agit de la carte Perle-1 développée au laboratoire Parisien de DEC [5] [6]. La carte Perle-1 comporte une matrice de 4×4 composants en logique programmable (FPGA Xilinx 3090), une mémoire statique rapide de 4 Moctets répartie tout autour de cette matrice et plusieurs autres FPGAs dédiés au contrôle. L'ensemble est connecté à une station de travail (DECstation 5000/240) via une interface TURBOchannel offrant une bande passante de 100 Moctets/sec.

Le réseau implanté sur la carte Perle-1 se compose de 128 processeurs. Il correspond, en fait, à une moitié de réseau par rapport à celui présenté dans le paragraphe précédent. La raison est que la taille des séquences étant, en général, très supérieure à la taille du réseau, un demi-réseau est beaucoup mieux adapté au partitionnement qu'un réseau complet.

La figure 3 donne le synoptique de l'architecture mise en oeuvre sur Perle-1. Les 128 processeurs sont connectés en un réseau linéaire unidirectionnel dans lequel les données (les caractères des séquences) transitent de la gauche vers la droite. Les données proviennent d'un module de décompression, noté DCP. Celles-ci sont, en effet, compressées pour limiter la bande passante entre la

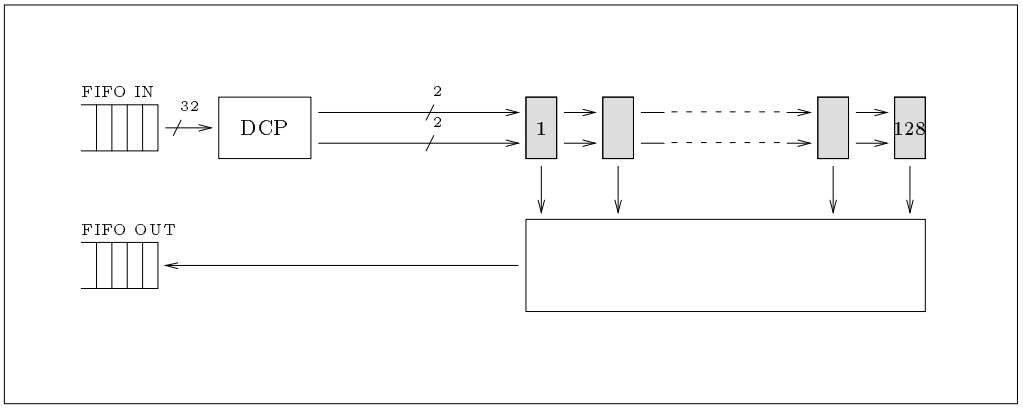

FIG. 3 - *synoptique de l'architecture implantée sur la carte Perle-1*

station de travail et le réseau ; on s'affranchit, ainsi, d'une mémoire locale de grande capacité à proximité du réseau.

La fonctionnalité d'un processeur est différente de celle mentionnée au paragraphe 2. Au lieu de calculer un maximum, un processeur détecte simplement que le score de la diagonale à dépassé un certain seuil S (donné par l'utilisateur). Ainsi, l'information délivrée pour un processeur est réduite à un seul bit. L'algorithme déroulé par un processeur est alors le suivant :

```
initialisation :
                    D = 0;
                    r = 0;

boucle systolique :
                    D = D + ((c0==c1) ? M : N );
                    D = max (D,0);
                    r = or (r,D>=S);
```

A l'issu d'un calcul, cette information est traitée globalement et est transformée en une série d'adresses correspondant aux numéros des diagonales dont le score est supérieur à S. Cette suite d'adresses est émise vers l'ordinateur hôte qui a ensuite pour tâche de recalculer précisément la valeur des scores.

D'un point de vue matériel, un processeur (ou une cellule systolique) est un élément relativement simple (cf figure 4) : une porte `XOR` détecte l'identité

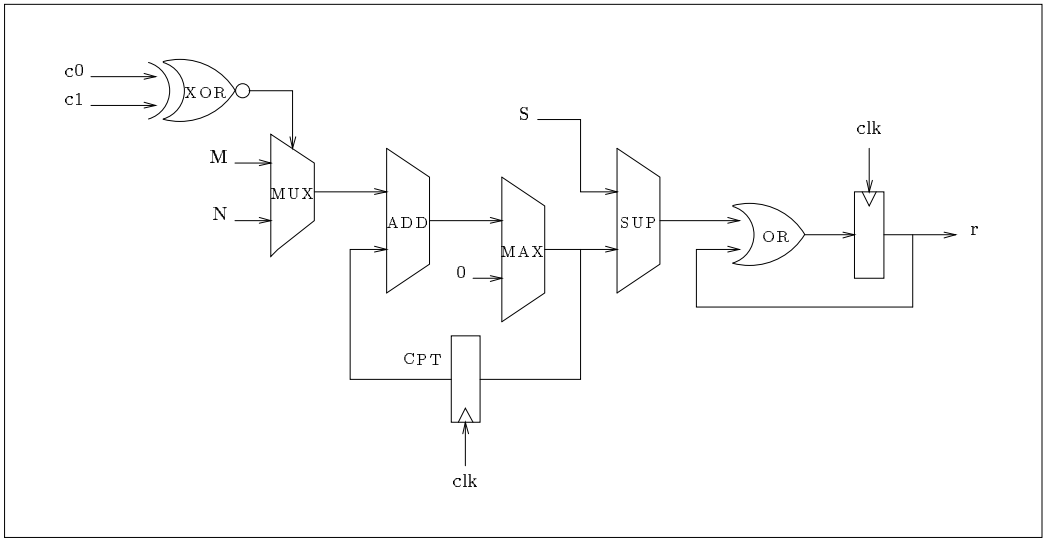

FIG. 4 - *architecture d'un processeur*

des deux caractères d'entrée (`c0` et `c1`). Suivant le résultat, un multiplexeur (`MUX`) sélectionne la valeur à ajouter (ou à retrancher) au compteur (`CPT`). La valeur du compteur ne peut devenir négative grace au maximum réalisé entre la sortie de l'additionneur et la valeur nulle. Le compteur est connecté à un comparateur (`SUP`) qui test si le score courant est supérieur à la valeur seuil S. Si tel est le cas, un indicateur (`r`) est positionné.

8 processeurs de ce type ont été implantés dans chacun des 16 Xilinx 3090 qui composent la matrice de la carte Perle-1, soit l'équivalent d'un réseau de 128 processeurs. Ce sont des processeurs sur 8 bits, ce qui limite la valeur du seuil S à 255. De plus, les valeurs `M` et `N` sont codées sur 3 bits; ce codage impose une dynamique réduite sur ces 2 paramètres, mais suffisante pour la majorité des hypothèses biologiques.

6 Expérimentations

Un logiciel, baptisé HScan, a été développé pour mener à bien différentes expérimentations. Comme BLASTN et FASTA, HScan produit d'abord une liste de séquences où ont été détectées des régions similaires. Puis il donne, sous forme d'alignement, toutes les paires de segments – entre la séquence test et les séquences de la banque – qui ont un score supérieur ou égal à un certain seuil (donné en paramètre). La structure du logiciel HScan est la suivante :

```
read (QS);                                  /* lecture sequence test */
NbSeqDB = open (DB);                       /* ouverture de la banque */
for (i=1;i<=NbSeqDB;i++)                    /* pour chaque sequence */
 {
   Read (DBS);                                /* lecture 1 sequence */
   NbDiag = filter(QS,DBS,M,N,T,NumDiag);     /* filtrage materiel */
   for (j=0;j<NbDiag;j++)
    {
     ComputeSSP(QS,DBS,NumDiag[j],M,N);     /* recalcul des scores */
    }
 }
```

Comme pour les logiciels BLAST et FASTA, le code est construit autour d'une boucle principale qui parcours l'ensemble de la banque. La procédure `filter()` correspond à l'émission vers le filtre systolique de la séquence test (`QS`) et de la séquence de la banque en cours de comparaison (`DBS`). Cette procédure donne en résultat le nombre de diagonales (`NbDiag`) dans lesquelles des zones similaires ont été détectées ; celles-ci sont répertoriées dans une table (`NumDiag`), ce qui permet à la procédure `ComputeSSP()` de recalculer précisément le score des segments similaires ainsi que leur situation par rapport aux deux séquences.

En pratique, la procédure `ComputeSSP()` est très peu exécutée. Par contre, la procédure `filter()` est systématiquement appelée ; c'est principalement elle qui détermine le temps d'exécution total de HScan.

Les expérimentations ont porté sur deux points :

1. comparaison des résultats entre le logiciel HScan et les deux logiciels : BLASTN et FASTA ;
2. mesure des performances en termes d'accélération du calcul.

6.1 Comparaison des résultats

La comparaison a été effectuée en prenant un lot de séquences test (30) sélectionnées arbitrairement dans GenBank, et en les confrontant avec une portion de cette même banque. Le résultat retenu est la suite de séquences qui possèdent des zones similaires avec la séquence test et les scores qui leur sont associés. Ne sont retenues que les séquences qui ont un score supérieur ou égal au seuil S calculé par BLASTN 1.4, c'est à dire les séquences qui contiennent des zones similaires jugées significativement intéressantes (non dues à la chance).

La référence est le résultat produit par HScan : l'exploration étant exhaustive, toutes les solutions sont détectées. La valeur des identités entre caractères (*match*) a été fixée à 4 ; celle des non-identités (*mismatch*) à -3. Aucune insertion ou omission n'est possible.

La sensibilité relative d'un logiciel est exprimée comme le rapport du nombre de séquences détectées par ce logiciel sur le nombre de séquences détectées par HScan. Ce calcul définit le pouvoir de détection du logiciel.

FASTA

La comparaison avec HScan a été effectuée en considérant les six valeurs possibles (1 à 6) du paramètre *ktup*. Rappelons que ce paramètre définit entièrement l'efficacité de la détection des zones de similarité. La valeur du score permettant de comparer les résultats entre les logiciels est celle référencée dans FASTA par *init1*.

Le diagramme de la figure 5 indique le pouvoir de détection (en %) de FASTA c'est à dire le nombre de séquences touvées en commun avec HScan pour les 6 valeurs du paramètre *ktup*.

Lorsque la sensibilté est maximale ($ktup = 1$), FASTA détecte 98 % des séquences détectées par HScan. Par contre, pour des sensibilités moindres, la

sensibilité de FASTA décroit rapidement. Moins de la moitié (48 %) des similitudes détectées par HScan ne le sont plus par FASTA lorsque le *ktup* est égal à 3. En règle générale, les fortes similitudes sont détectées quelle que soit la valeur du paramètre *ktup* ; par contre, celles qui ont un score proche du seuil ne sont plus détectées dès lors que la valeur du *ktup* dépasse 3 ou 4.

BLASTN 1.3

La sensibilité de BLASTN 1.3 ne peut pas être ajustée. Plus exactement, le seul moyen mis à la disposition de l'utilisateur est la modification de la taille du mot (groupement de plusieurs caractères) sur lequel s'effectue les comparaisons. Par défaut cette taille est fixée à 12 ; toute modification entraine un avertissement spécifiant qu'en dehors de cette valeur le comportement du logiciel n'est pas garanti !

Le diagramme de la figure 5 indique le pouvoir de détection (en %) de BLASTN 1.3 employé en usage normal (taille de mot fixée à 12).

Il est clair que BLASTN 1.3 trouve relativement peu de solutions (13 %) par rapport à HScan. Son pouvoir de détection est du même ordre de grandeur que FASTA utilisé avec un *ktup* de 5 ou 6 ; les séquences qui présentent des zones à forte similarité sont systématiquement trouvées. L'analyse de l'algorithme sous-jacent a montré qu'une zone ne pouvait être candidate que si elle présentait au moins 12 identités (caractère à caractère) consécutives. Cela revient à une recherche par BLASTN 1.4 avec une taille de mot fixée à 12. La sensibilité entre les deux cas est d'ailleurs strictement égale. Or, il existe de nombreux cas où des paires de segments sont similaires et qui ne partagent pas une suite de 12 caractères identiques. On peut noter que c'est d'ailleurs la raison principale qui a motivé la modification de BLASTN 1.3 en une nouvelle version, BLASTN 1.4, dans laquelle la taille du mot W est paramétrable.

BLASTN 1.4

La sensibilité de BLASTN 1.4 peut être ajustée et la comparaison a été effectuée en considérant 7 valeurs (1 à 6 et 12) du paramètre W (taille du mot). Ce choix permet à la fois de comparer les résultats avec FASTA (*ktup* 1 à 6), BLASTN 1.3 (W=12) et avec HScan.

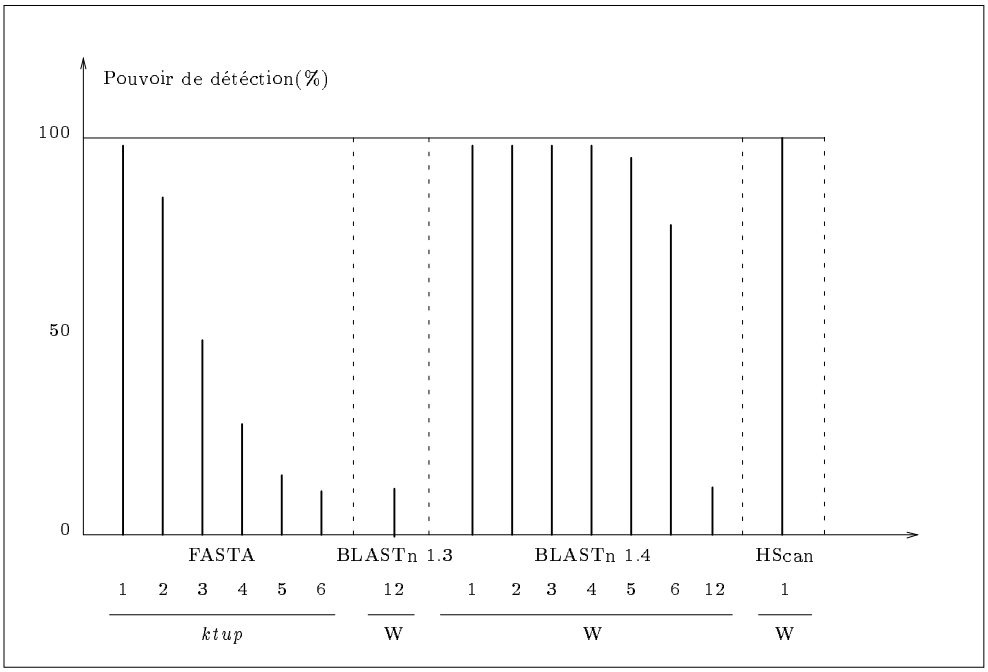

FIG. 5 - *Variation du pouvoir de détection de différents logiciels en fonction de la taille du mot. La référence est* HS*can*

Le diagramme de la figure 5 indique le pouvoir de détection (en %) de BLASTN 1.4 pour les 7 valeurs choisies du paramètre W.

Comme pour FASTA, BLASTN 1.4 détecte 99 % des séquences trouvées par HScan. Par contre la dynamique de diminution de sensibilté en fonction de la variable de taille du mot (*ktup* ou W) est beaucoup plus lente avec BLASTN 1.4 qu'avec FASTA. Par exemple pour une même taille de 5 caractères, 97% des séquences sont encore détectées par BLASTN 1.4 tandis que seulement 16% le sont encore avec FASTA. Cette différence de cinétique entre les deux logiciels devra être prise en compte pour le choix des options de départ en fonction du problème biologique posé.

L'objectif d'exhaustivité des résultats dans l'utilisation de HScan est atteinte. Toutes les séquences détectées par les logiciels standards utilisés en sensibilité maximale (recherche caractère par caractère) le sont par HScan. Certaines séquences sont même trouvées par HScan uniquement.

Ce dernier point mérite d'être approfondit. Dans le cas d'une recherche caractère par caractère (*ktup*=1 pour FASTA et $W = 1$ pour BLASTN 1.4) les résultats devraient être identiques à ceux trouvés par HScan. En fait, BLASTN 1.4 opère un deuxième niveau de filtrage : les segments trouvés, même s'ils possèdent un score supérieur ou égal au seuil fixé, doivent, en plus, répondre à un critère probabiliste pour être proposés comme résultats. Certaines séquences sont donc éliminées par cette deuxième contrainte ; si celle-ci n'existait pas, les résultats seraient identiques.

Dans le cas de FASTA, la différence s'explique par l'algorithme de recherche mis en oeuvre. La détection des segments similaires est réalisée suivant le principe de l'algorithme présenté au début de ce rapport, mais diffère sur un point : au lieu de remettre à zéro la valeur d'un score lorsqu'il devient négatif, FASTA peut autoriser d'abaisser le seuil suivant des critères dépendant des données en cours de traitement. Ainsi, partant d'une valeur inférieure à zéro, le score final est forcément plus faible qu'un score calculé en prenant la valeur nulle comme point de départ.

6.2 Accélération du calcul

L'accélération d'un calcul peut être mesurée comme le rapport entre le temps d'exécution produit par deux implémentations fournissant des résul-

tats identiques. Dans notre cas, c'est le rapport entre le temps d'exécution de BLASTN 1.4 ou de FASTA (version 1.7) sur le temps d'exécution de HScan. Cette mesure a été effectuée en positionnant, dans les 2 cas, la taille des mots à 1 de manière à obtenir des résultats équivalents à HScan.

Les courbes de la figure 6 montrent les accélérations obtenues par rapport à l'exécution de BLASTN et FASTA sur deux stations de travail : une SPARC 2 et une SPARC 10. La puissance de calcul de la première machine est équivalente à celle des meilleurs PC actuels (en 1995) tandis que la seconde machine possède les performances typiques d'une bonne station de travail. Ces mesures ont été effectuées pour des séquences test de taille variable et constituent une moyenne ; en effet, la composition des séquences test influe sur les temps de calcul des logiciels BLASTN et FASTA.

De manière générale, on constate que plus la séquence test est longue, meilleure est l'accélération. Cette caractéristique est intéressante pour l'exploitation future de ce type de matériel dans la mesure où les nouvelles techniques de biologie moléculaire permettent (et permettront) d'obtenir des séquences de plus en plus longues.

Les performances de HScan, d'un point de vue vitesse d'exécution, sont grosso modo comparables à celles de BLASTN 1.3 s'exécutant sur un PC, c'est à dire qu'il faut environ quelques dizaines de secondes pour explorer entièrement une banque de séquences d'ADN. A temps de calcul identique, HScan obtient néanmoins des résultats beaucoup plus complets (cf paragraphe précédent).

Un deuxième type de mesure a été réalisé pour étudier l'influence de la taille du filtre sur le parallélisme effectif. Les temps de calcul ont donc été mesurés en faisant varier le nombre de processeurs de 32 jusqu'à 128, et en prenant comme base le temps d'exécution sur un réseau de 32 processeurs.

La figure 7 montre globalement que plus la taille de la séquence test est longue, meilleur est le parallélisme. Autrement dit, un filtre de grande taille est mal exploité lorsque de courtes séquences doivent être confrontées aux banques de données. Par contre, dès que la taille de la séquence test atteint quelques milliers de nucléotides, un filtre de grande taille contribue à de meilleurs performances.

Ces résultats rejoignent les courbes d'accélération mesurées précedemment et confortent notre vision des structures de filtres à élaborer pour traiter les

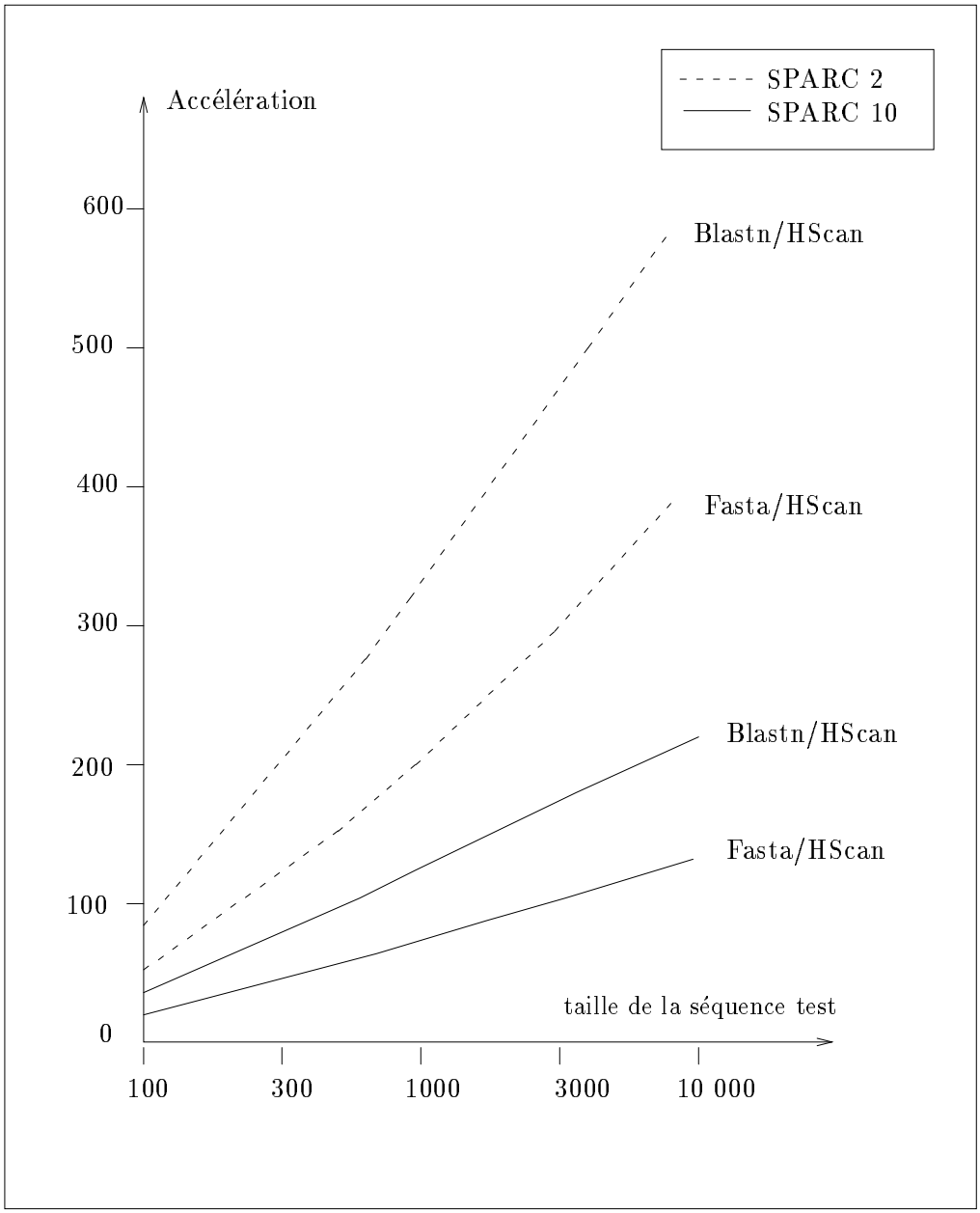

FIG. 6 - *accélération mesurée comme le rapport du temps de calcul de* BLASTN *et*FASTA *sur le temps de calcul de* HS*can. Les deux droites représentent l'accélération moyenne obtenue sur un nuage de points.*

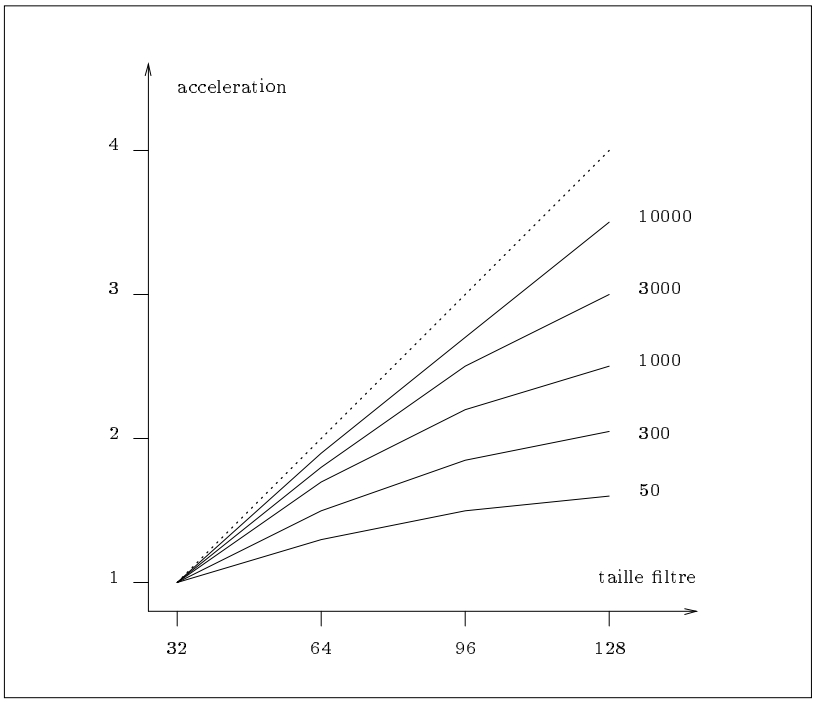

FIG. 7 - *parallélisme obtenu en fonction de la taille du filtre sur un échantillon de séquences test de différente longueur (50, 300, 1000, 3000 et 10 000)*

séquences biologiques de demain : plus leur taille sera grande, meilleur sera le temps de réponse des logiciels qui les inclueront.

7 Comparaison par rapport aux autres systèmes

Plusieurs études et réalisations matérielles ont été proposées pour réduire le temps d'exécution induit par la comparaison des séquences biologiques. Le but de ce paragraphe est de faire un rapide tour d'horizon sur les différentes propositions pour mieux situer, par la suite, l'originalité de notre approche.

Parmi toutes les solutions proposées, nous présentons seulement celles qui se rapprochent le plus de nos travaux, c'est à dire les approches matérielles dédiées et non programmables (voir [3] pour une étude plus détaillée). Globalement, deux possiblités ont été étudiées : la réalisation de machines spécialisées à bases de circuits intégrés spécifiques et l'implémentation d'algorithmes en logique reconfigurable [15].

Les machines spécialisées

En général, les machines spécialisées sont constituées d'un réseau comprenant un grand nombre de processeurs câblés (i.e. non programmables). Ces processeurs étant peu complexes, un circuit intégré peut en contenir plusieurs. L'assemblage de plusieurs circuits constitue le réseau.

La machine BioScan [16] appartient à cette famille. Elle a été conçue autour d'une puce qui intègre environ 800 processeurs 1-bit connectés en une structure linéaire. Le tout occupe une surface de 72 mm^2 (CMOS 1,2 micron). Il est bien sûr possible de cascader plusieurs puces pour obtenir un réseau de taille supérieure.

La réalisation actuelle est basée sur une carte VME (connectée à une station de travail de type SUN-4) pouvant accueillir entre 10 et 20 circuits. Les banques de séquences transitent par le bus VME et sont régulées par une file d'attente à l'entrée du réseau. La carte supporte également un mécanisme de décompression/formattage de donnée et une unité de contrôle permettant la gestion bit série de l'ensemble.

L'algorithme implanté dans le silicium permet la détection de segments similaires suivant la définition de BLASTN. Les performances de cette machine permettent, par exemple, d'explorer en totalité GenBank 77 (139 millions de bases) en 71 secondes. Le système est mis à la disposition de la communauté scientifique par serveur accessible par ftp ou e-mail.

La machine BISP (*Biological Information Signal Processor*) [7] fait également partie de cette famille. Un prototype de 256 processeurs a été réalisé et validé. Le système complet comporte une interface programmable (processeur 68020) qui a la charge de gérer les transferts avec l'hôte, d'alimenter le réseau en données et de collecter les résultats. C'est donc une machine autonome qui, lorsqu'elle est sollicitée, gère entièrement un traitement et produit une liste de résultats.

L'algorithme implémenté est un algorithme de programmation dynamique. Les informations délivrées sont donc de nature différente de celles fournies par BioScan. Il est en effet possible, de calculer des alignments où l'insertion et l'omission de caractères est autorisé.

La machine SAMBA (*Systolic Accelerator for Molecular Biological Applications*) développée par notre équipe [3] possède un réseau quasiment identique à celui de la machine BISP (256 processeurs) et supporte pratiquement les mêmes fonctionalités. Elle diffère principalement par le couplage avec la station hôte réalisé en logique reprogrammable.

Ces deux dernières machines (BISP et SAMBA) ont une vocation beaucoup plus générale que l'accélérateur matériel que nous présentons dans ce document. Elles peuvent traiter aussi bien des séquences d'ADN que des séquences protéiques avec des algorithmes plus complexes.

Les machines à base de logique reconfigurable

On peut recenser deux machines de ce type, conçues principalement pour accélérer des algorithmes d'analyse de séquences biologiques. Il s'agit des machines Bioccelerator et SPLASH-2. Le coeur de ces machines est constitué de circuits logiques reconfigurables (ou FPGA : *Field Programmable Gate Array*) [15].

Un FPGA est un composant électronique composé d'une matrice de blocs élémentaires dans lesquels une fonction logique peut être programmée. Ce

composant dispose de ressources de routage (également programmables) permettant d'associer ces blocs élémentaires. Ainsi, l'élaboration d'un opérateur arithmétique – un additionneur 16 bits par exemple – consiste à déterminer la fonction logique binaire de l'addition dans 16 blocs élémentaires, puis à spécifier les connexions entre ces blocs. Une architecture matérielle est constituée d'un assemblage d'éléments spécifiés de la sorte. La programmation d'un tel composant est entièrement dynamique et s'effectue en quelques millisecondes.

La machine Bioccelerator [8], développée au Weiztmann Institute of Science, en Israël, est dédiée à l'accélération de certains programmes du progiciel GCG (Genetics Computer Group) [10] [9] . Elle peut comprendre de une à quatre cartes contenant chacune quatre noeuds de calcul. Un noeud contient un FPGA Xilinx 4008 [17] et une mémoire statique rapide.

Cette machine n'est donc pas dédiée à un algorithme particulier : en modifiant les configurations associées aux FPGAs, sa structure peut être adaptée à un grand nombre d'algorithmes. Bioccelerator est commercialisée et constitue une machine autonome accessible par réseau. Elle accélère, entre autres, les programmes Fasta, Tfasta, Pileup, ProfileSearch, ProfilScan ProfilSegment et GelMerge du progiciel GCG.

La machine SPLASH-2 [2], conçue au SRC (*Supercomputing Research Center - Institute for Defense Analyses*) est le successeur de la machine SPLASH-1 [13] initialement conçue pour la comparaison de séquences biologiques.

SPLASH-2 est un réseau linéaire dont chaque noeud est composé d'un FPGA (Xilinx 4010, l'équivalent de 10 000 portes logiques) et d'une mémoire statique de 512 Ko. La configuration maximum est de 256 noeuds répartis sur 16 cartes distinctes. L'interfaçage avec une station de travail (Sun Sparc Station) est assurée par le SBus (bus spécialisé dans les entrées/sorties) et autorise un débit maximum de 54 Mo/s.

L'intérêt de SPLASH-2, tout comme la machine Bioccelerator est de pouvoir *programmer* une infinité d'architectures. Les premières mises en oeuvre d'algorithmes d'analyse de séquences [12] montrent qu'il est possible d'implanter des réseaux de processeurs conséquents, permettant ainsi d'accélérer ces applications dans de fortes proportions.

Notre approche

Notre approche se démarque des autres réalisations par le couplage extrêmement fort qui existe entre le logiciel et le matériel. La fonction du filtre est simplement de repérer des zones de similitude potentielle ; à partir de ces informations, le logiciel (ou plus exactement le processeur de la machine hôte) localise précisément ces zones et calcule un score fonction de la ressemblance. A l'inverse, toutes les autres réalisations déterminent matériellement le score exact et les positions des alignements.

Ce filtre agit plutôt comme un co-processeur alors que les machines BioScan, Bisp, Samba, Bioccelerator ou Splash s'utilisent comme des unités périphériques autonomes auxquelles on confie une tâche – comparer, par exemple, telle séquence contre telle ou telle banque – et qui délivrent une série de résultats définitifs. Cette différence fondamentale s'exprime à deux niveaux :

- la complexité des processeurs (ou cellules systoliques) n'est pas la même dans les 2 cas. D'un coté on a besoin de processeurs pouvant effectuer des calculs sur une dynamique importante (au moins 16 bits pour couvrir l'ensemble de la classe des problèmes) alors que de l'autre coté, les calculs sont réduits à leur plus simple expression.

- la complexité de l'interface entre le réseau et l'hôte est également différente. En fait, les machines autonomes (sauf SPLASH et SAMBA) ont recours à des processeurs programmables (spécialisés ou non) qui se chargent du contrôle du réseau, des problèmes de partitionnement, de l'accès aux données, etc. Dans notre cas, c'est directement le processeur hôte qui effectue ces tâches : les puissances de calcul dispensées par les microprocesseurs d'aujourd'hui conjugées à des bus d'entrées/sorties à bande passante élevée, permettront de plus en plus de s'affranchir de tels dispositifs intermédiaires.

8 Conclusion et perspectives

L'accélérateur matériel proposé a pour but de pallier les inconvénients des logiciels d'exploration des banques de séquences d'ADN. Ceux-ci sont de deux ordres : non exhaustivité des solutions proposées et lenteur du calcul. Une

approche purement logicielle permet soit une exploration rapide mais avec des résultats partiels (BLASTN 1.3), soit une détection exhaustive mais avec un temps de calcul important (FASTA et BLASTN 1.4. La substitution de la phase de détection des zones de similarité potentiellement intéressantes par un filtre matériel permet de s'affranchir simultanément de ces deux inconvénients. Le temps de calcul obtenu est alors de l'ordre de celui obtenu avec BLASTN 1.3 pour une sensibilité égale, voire supérieure à celle de FASTA ou BLASTN 1.4 utilisés en sensibilité maximale.

La méthode de détection proposée est suffisamment générale pour que l'intégration d'un filtre matériel à des logiciels existants puisse être réalisée. Une étude dans ce sens à d'ailleurs été réalisée sur FASTA et a montré qu'un facteur d'accélération de 50 à 100 pouvait être obtenu par rapport à la version purement logicielle. Cette expérimentation est décrite dans l'annexe 2.

D'un point de vue matériel, plusieurs implémentations sont envisageables. La première consiste à poursuivre celle qui a été réalisée dans le cadre de cette étude, c'est à dire à programmer sur un support à base de logique reconfigurable un réseau de processeurs relativement simples. La carte Perle-1, si elle est très bien adaptée aux expérimentations que nous avons pu mener, est trop *riche* pour cette application : les mémoires rapides qui couronnent la matrice de FPGA ne sont pas utilisées et les circuits FPGA annexes à la matrice sont largement sous-exploités.

Une carte plus simple, composée seulement de quelques composants FPGA de technologie récente (Xilinx série 4000 ou 5000) connectés en anneau, permettrait d'atteindre des performances identiques, voire supérieures. Une telle carte pourrait directement s'insérer dans les slots d'extension d'un PC ou d'une station de travail. L'avantage de cette solution est le caractère reconfigurable du matériel : on peut, en effet, imaginer que d'autres problèmes biologiques similaires – dans le domaine de la comparaison de séquences en particulier – puissent être traités de façon similaire, notamment pour l'exploration de banques protéiques.

Une seconde possibilité consiste à réaliser une puce entièrement dédiée à ce calcul. L'avantage est que la faible complexité des processeurs permet d'en intégrer un grand nombre sur une seule unité. Une évaluation succinte a montrée que pour une technologie courante (CMOS 0,8 micron), et sur une surface raisonnable ($<1\text{cm}^2$), un réseau de 256 processeurs peut aisément être intégré.

Par rapport à la solution précédente, celle-ci a l'avantage d'être plus compacte ; par contre elle ne permet qu'un seul type de traitement.

Par rapport aux machines spécialisées existantes, notre approche se distingue avant tout par un volume matériel réduit et par une interaction de type co-processeur entre le processeur de la machine hôte et le filtre systolique. Pour ce traitement particulier, cet accélérateur accroit fortement – et à faible coût – les performances d'une machine standard au même titre, par exemple, qu'un co-processeur vidéo pour le traitement d'images.

Remerciements

Nous tenons à remercier Patrice Quinton, Francois Bodin et Irène Wang pour la lecture attentive de ce rapport et leurs nombreuses suggestions qui, nous l'espérons, contribuent à la clarté et à la compréhension de ce texte.

Références

[1] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. Basic local alignment search tool. *J. Biol. Mol*, (215):403–410, 1990.

[2] J.M. Arnold, D.A. Buell, and E. G. Davis. SPLASH 2. In *4th Annual ACM Symposium on Parallel Algorithms and Architecture*, 1992.

[3] L. Audoire, JJ. Codani, D. Lavenier, and P. Quinton. Machines spécialisées pour la comparaison de séquences biologiques. *TSI*, 14(1):9–22, January 1995.

[4] D. Benson, D. J. Lipman, and J. Ostell. Genbank. *Nucl. Acids Res.*, 21(13):2963–2965, 1993.

[5] P. Bertin. *Mémoires actives programmables: conception, réalisation et programmation.* PhD thesis, Université Paris 7, juin 1993.

[6] P. Bertin, D. Roncin, and J. Vuillemin. Programmable active memories : a performance assessment. In F. Meyer auf der Heide, B. Monien, and A.L. Rosenberg, editors, *Parallel Architectures and their efficient use*,

pages 119–130, Lecture notes in Computer Science, Springer-Verlag, oct 1992.

[7] E. Chow, T. Hunkapiller, and J. Peterson. Biological Information Signal Processor. In *ASAP*, pages 144–160, sep 1991.

[8] Compugen. *The BIOCCELERATOR machine*. Israel, 1993.

[9] J. Devereux and al. A comprehensive set of sequence analysis programs for the vax. *Nucl. Acids Res.*, 12:387–395, 1984.

[10] *Program Manual for the GCG package, version 7*. Genetic Computer Group, 575 Science Drive, Madison, Wisconsin, USA 53711, Apr 1993.

[11] R. Halfhill. 80x86 wars. *Byte*, 74–88, June 1994.

[12] D. T. Hoang. Searching Genetic DataBases on SPLASH-2. In D.A. Buell and K.L. Pocek, editors, *FPGAs for custom computing machines*, pages 185–191, IEEE Computer Society Press, apr 1993.

[13] D. Lopresti. Rapid Implementation of a Genetic Sequence Comparator Using FPGAs. *Advance Research in VLSI 1991*, 139–152, 1991.

[14] W. R. Pearson and D.J. Lipman. Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci.*, 85:3244–3248, 1988.

[15] J. Rose, A. El Gamal, and A. Sangiovanni Vincentelli. Architecture of Field-Programmable Gate Arrays. *Proceedings of the IEEE*, 81(7):1013–1029, jul 1993.

[16] C.T. White, R.K. Singh, P.B. Reintjes, J. Lampe, B.W. Erickson, W.D. Dettloff, V.L. Chi, and S.F. Altschul. BioSCAN: A VLSI-Based System for Biosequence Analysis. In *IEEE Int. Conf on Computer Design: VSLI in Computer and Processors*, pages 504–509, IEEE Computer Society Press, oct 1991.

[17] Xilinx. *The XC4000 data book*. The programmable logic company, 2100 Logic Drive, San Jose, CA 95124, aug 1992.

Annexe 1
architectures et algorithmes systoliques

Le terme systolique est une métaphore qui provient de la similarité dans le fonctionnement de ce type d'architecture avec le travail régulier du coeur. Une architecture systolique est un réseau composé d'un grand nombre de processeurs simples, ou cellules, qui réalisent simultanément les mêmes opérations élémentaires sur un flot de données continu. Chaque cellule reçoit les données en provenance des cellules voisines, effectue un calcul, puis transmet les données et ses résultats à des cellules voisines un temps de cycle plus tard. Les cellules sont interconnectées localement, seules les cellules situées aux extrémités du réseau communiquent avec le monde extérieur, c'est à dire une mémoire externe ou un ordinateur hôte.

Les flots de données se propagent à travers le réseau à vitesse constante, suivant des chemins réguliers, et interagissent dans les processeurs la où ils se rencontrent. Les calculs réalisés par chaque processeur sont peu complexes et exécutés de manière synchrone. Un algorithme systolique contient donc à la fois la séquence d'opérations exécutée par les processeurs et le mouvement des données entre les processeurs.

exemple

Considérons l'algorithme systolique qui consiste à évaluer un polynome suivant le schéma de Horner. Ce dernier utilise la décomposition suivante :

$$y_i = c_0 + c_1 x_i + c_2 {x_i}^2 + ... + c_d {x_i}^d = c_0 + x_i(c_1 + x_i(c_2 + x_i(.... + x_i c_d)))$$

La figure 8 illustre une mise en oeuvre systolique de cet algorithme.

Un coefficient du polynome est alloué à chaque processeur d'un réseau linéaire. Les valeurs x_i sont injectées par la gauche du réseau et propagées vers la droite au même rythme que les résultats partiels. Un processeur $P(j)$ reçoit la donnée x_i et la valeur partielle calculée p_{j-1} par son voisin de gauche. Il réalise le calcul suivant : $x_i \times p_{j-1} + coef$ et le propage à son voisin de droite avec la valeur x_i.

Le résultat final du calcul de y_i est délivré à l'étape $i+d$ par le processeur de droite. L'ensemble des opérations qui sont effectuées à cahque étape du calcul constitue ce qu'on appelle un *cycle systolique*. La figure 8 représente l'état du réseau après le quatrième cycle.

Un processeur ne participe qu'une fois au calcul d'un polynôme mais il est disponible pour effectuer un calcul sur une nouvelle donnée au cycle suivant. Après le temps de latence correspondant à la propagation de la première donnée à travers le réseau, le régime de calcul optimal est atteint : le réseau délivre un résultat par cycle.

Pour fonctionner correctement, le réseau doit être alimenté à chaque cycle avec les valeurs x_i et la constante 0, qui correspond à la valeur initiale de y_i. A l'autre extrémité, il faut récupérer les valeurs de y_i sauf pendant les $d+1$ premiers cycles de calcul.

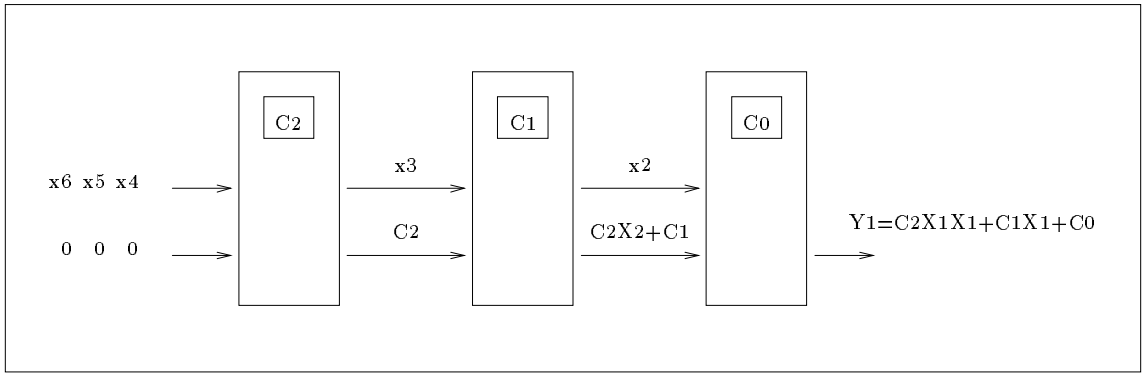

FIG. 8 - *réseau systolique linéaire pour le schéma de Horner*

Annexe 2
intégration du filtre dans FASTA

Le logiciel que nous avons développé pour nos besoins expérimentaux, HScan, est plus proche de BLASTN que de FASTA dans la mesure où il ne donne en résultats que des alignements de segments ayant des longueurs identiques. FASTA essaie, au contraire, de regrouper plusieurs segments en introduisant des *gaps* (insertion ou omission de caractères) pour former des segments plus longs et pas nécessairement de même longueur. Le temps de calcul de cette seconde phase étant faible par rapport au temps d'exécution total, il nous a paru intéressant de substituer la phase de détection logicielle de FASTA par le filtre matériel afin d'améliorer les performances globales du traitement.

Si le remplacement immédiat de la phase de détection par l'accélérateur matériel apporte un gain temporel appréciable (entre 50 et 100), il ne conduit pas, par contre, aux mêmes résultats. Plus précisement, les résultats sont identiques lorsque de fortes similitudes sont mises en évidence ; ils deviennent dissemblables dès que l'indice de ressemblance atteint un seuil partagé par un grand nombre de séquences.

Cette différence s'explique principalement par la sélection arbitraire des séquences qui est effectuée, dans FASTA, pendant la phase de filtrage : la recherche de similitude entre la sequence test et une séquence de la banque est basée sur la mémorisation des dix diagonales qui ont produit les meilleurs scores. Par conséquence, les diagonales ayant un score élevé sont systématiquement retenues alors que celles qui présentent des scores plus faibles – et qui sont en compétition à cause de l'égalité de leur score – sont choisies arbitrairement en fonction de l'algorithme mis en place.

Deux versions de FASTA "accéléré" ont donc été testées. La première exécute, après filtrage matériel, une partie du code utilisé pendant la détection ; ce traitement a simplement pour but de réorganiser les données pour les phases suivantes. On obtient ainsi des résultats identiques à ceux de la version originale. La seconde version n'inclut pas cette remise en forme des données.

D'un point de vue biologique, les deux versions se valent : lorsque les similitudes sont indiscutables, le classement et les scores sont rigoureusement identiques. Par contre, lorsque les scores indiquent que la frontière entre une

similitude probable et une similitude due au hasard est atteinte des différences notables apparaissent. La seconde version a, bien sûr, l'avantage d'être plus rapide.

Cette expérimentation montre que l'accélérateur matériel que nous proposons est suffisamment souple pour être introduit dans des logiciels existants. Cette caractéristique est importante : elle permet de s'affranchir des réticences qu'il est courant de rencontrer lorsque de nouveaux logiciels sont proposés. L'ajout du filtre ne donne pas lieu à des des modifications qui nécessite un nouvel apprentissage de l'outil. La différence s'exprime seulement sur le temps de réponse, beaucoup plus court.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399